COMPASS Note 1998-16

CATCH1

A Test-facility for COMPASS Front-End Electronics

User Manual

G. Braun, H. Fischer, J. Franz, A. Grünemaier,
F.H. Heinsius, K. Königsmann, M. Schierloh,
T. Schmidt, H. Schmitt, J. Urban

Fakultät für Physik, Universität Freiburg, 79104 Freiburg, Germany

November 23, 1998

Abstract

A 6U VME based printed circuit board for testing the functionality of the COMPASS front-end electronics will be described. The user can generate up to 64 independent test pulses to simulate pulses from a detector system. The data produced from these artificial signals (or from a real detector equipped with front-end boards) can be read out to a computer via the VMEbus. Readout and control of the front-end boards is done via a standard RJ-45 connector, as used in COMPASS. A flexible design allows to create complex test-setups.

Contents

1 Introduction **3**

2 Functional Description **5**
2.1 VME Interface 6
2.2 HOTLink serial receiver 8
2.3 FIFO Buffer 9
2.4 Serial Transmitter (10 Mbit/s) 10
2.5 Test Pulse Controller 10
2.6 User Command Coding Unit 13

3 Board Layout **16**
3.1 Front Panel Information 16
3.2 Front Panel Pin Assignment 16
3.3 Rear Panel Pin Assignments 19
3.4 Jumper Settings 19
3.5 Rotary Switches 20

4 Components on the Board **21**
4.1 Xilinx XC95288-15 CPLD 21
4.2 Xilinx XC4020E-2 FPGA 21
4.3 Cypress CY7B933 HOTLink Receiver 22
4.4 CY7C4251-15 JC FIFO 22
4.5 Lattice GAL22V10C GAL 22
4.6 CY7C199-10vc Memory 22

5 Electrical and Mechanical Specifications **23**
5.1 Cables 23
5.2 Signal Characteristics 23
5.3 Power Requirements, Mechanical Size 25

A Getting started with the CATCH1 **26**

B Example: FPGA Configuration **27**

C List of Sheets **30**

1 Introduction

Experiments in modern high energy physics have a demand for higher precision and therefore feature an increasing number of readout channels of the detectors.

For the COMPASS experiment at CERN which is a fixed target spectrometer allowing beam intensities up to $2{\cdot}10^8$ particles per spill, new electronics for digitization and a new data acquisition system is being developed which can cope with the large data rates.

Once the new front-end electronics reach the stage of mass-production, they will have to be tested thoroughly before being used in the experiment. The CATCH1-module is the ideal facility for testing purposes. It features the same communications with the front-end as those being used in the COMPASS experiment. All the data between front-end board and CATCH1 are transferred via an S-UTP cable (CAT 5+): the digitized detector information (Chapters 2.2 and 2.3), initialization data for the front-end electronics (Chapter 2.4), four USER commands (e.g. TRIGGER) for the front-end (Chapter 2.6) and distribution of a 40 MHz clock. The main difference is that the CATCH1 only communicates with one front-end board (64 channels) whereas in the experiment 16 front-end boards will be read out by one CATCH-module.

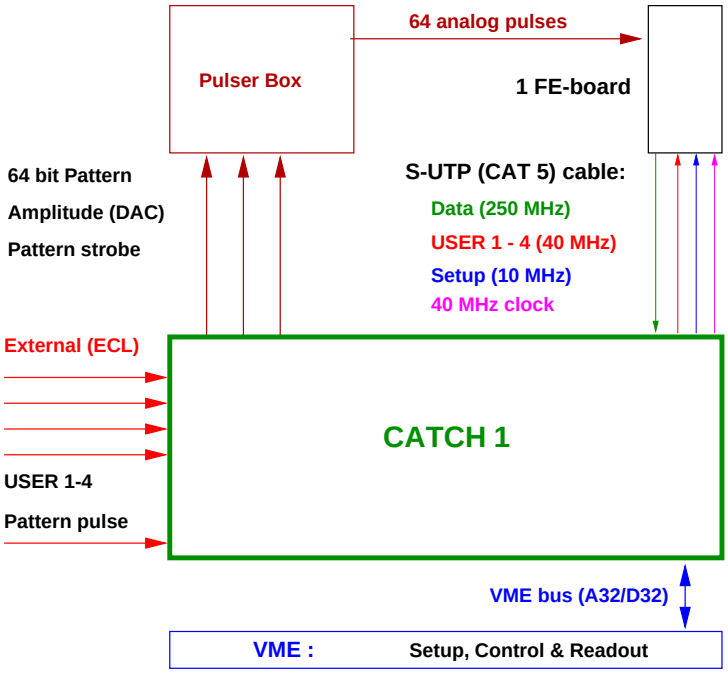


Figure 1: Possible test-setup with the CATCH1 board

The CATCH1 is a very flexible device. It provides up to 64 independent TTL test pulses which can be programmed via the VMEbus. The pulses are converted into analog signals in an external 'Pulser Box[1]' before being used to simulate a detector system. This makes it easy for the user to adjust pulse shapes to the needs of the front-end system which is to be tested. The data flow for a possible test-setup can be found in Figure 1.

Another feature are internal scalers for the USER commands (Chapter 2.5). This, for instance, allows to keep track of the number of external TRIGGER signals. All functions of the CATCH1 board as well as the readout of the front-end data are controlled via the VMEbus (Chapter 2.1). Additional connectors on the front panel allow to apply external signals as well.

In the experiment the VME interface will only be used to transfer data needed for initialization of the front-end electronics and to collect data for slow control purposes. The data from the front-end side will be ordered by events on the CATCH-module before being passed ahead to the on-line filtering via a fast optical S-Link.

[1]PULPO: PULse and Pattern bOx

2 Functional Description

The CATCH1 is a one unit wide 6U VME slave-module. Its main features are a unit to control an external module for test pulse generation, a USER Command Coding Unit, one serial port to transmit data, one to receive data ('HOTLink') and the VME interface logic. See the block diagram (Figure 2) for more details.

The Test Pulse Controller receives test pattern and four USER commands (in the following referred to as Trigger, Reset, Clear, User) from the VMEbus. This unit can apply 64 independent TTL test pulses to a connector (80 pins) on the front panel of the CATCH1. Via a flat cable the digital signals are transferred to an external device ('Pulser Box') which converts them into analog signals. The control signals for pulse height selection and the test pulse duration are also transferred via this cable. Moreover, for the USER commands, the Test Pulse Controller provides scalers which can be read out via the VMEbus.

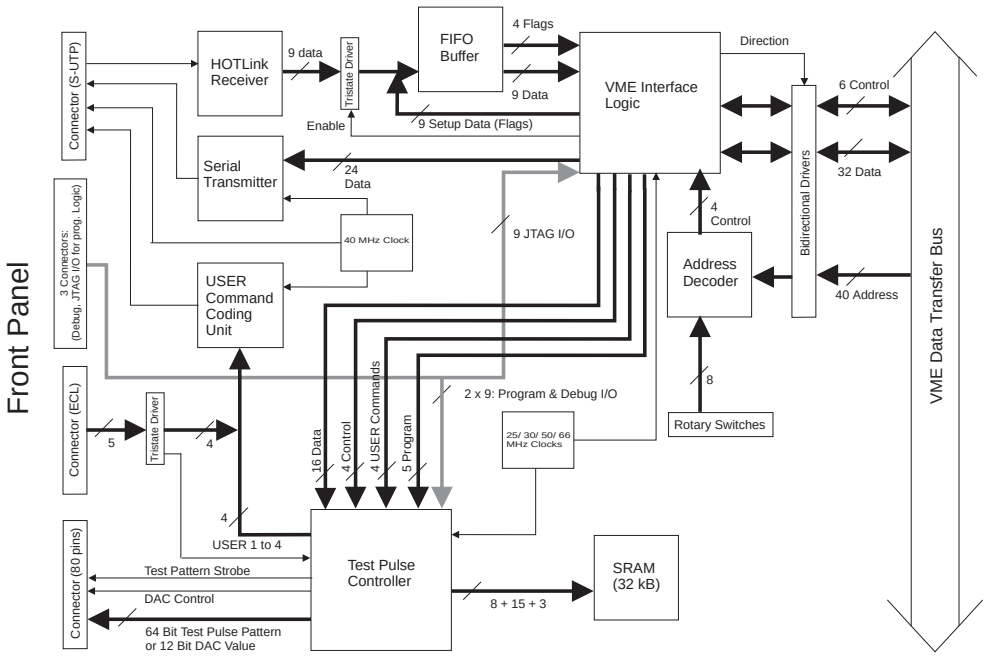


Figure 2: Block diagram of the CATCH1 board

The four different USER commands, which can also be applied as ECL level signals on the front panel, are encoded into signals of different lengths

by the User Command Coding Unit before being transmitted to the front-end boards via one pair of wires on a S-UTP cable (Cat 5+).

The serial transmitter provides the possibility to transfer data needed for initialization of the front-end electronics. This serial port runs at 10 Mbit/s and transfers 24 bit of data on a second pair of wires on the S-UTP cable with each transmission. It cannot receive data and has no provision for error handling.

The third wire-pair on the S-UTP cable is used to transmit the 40MHz clock to the front-end. On the CATCH1 this clock is used by the User Command Coding Unit and the serial transmitter. Both components send their data synchronously to the clock.

Data generated on the front-end boards are received by the HOTLink via the last pair of wires on the S-UTP cable. It operates at frequencies up to 330 MHz and monitors transmission errors. Having arrived on the CATCH1-module the data are parallelized into words of 8 bit. The data are stored in a FIFO buffer from where they can be read out to the VMEbus. An additional bit indicating transmission errors is also stored in the FIFO buffer being nine bit wide.

2.1 VME Interface

The VME interface of the CATCH1 consists of an address decoder and a state machine. The address decoder monitors the VME address lines and passes a valid command to the state machine where the corresponding actions are carried out. Table 1 lists the VME data transfer bus signals monitored by the CATCH1.

Table 1: VME data transfer bus signals

Addressing	Data	Control
A31-A01	D31-D00	AS*
AM5-AM0		DS0*
DS0*		DS1*
DS1*		WRITE*
LWORD*		(BERR*)
		(DTACK*)
		SYSRESET*

DTACK* is asserted by the module itself and the signal BERR* is foreseen but not yet implemented in the design. The CATCH1 works in the A32

addressing mode and recognizes the following Address Modifier codes (a "%" indicates hexadecimal numbers):

AM0-AM5 = %0D	extended supervisor data access
AM0-AM5 = %09	extended non-privileged data access

The base address of the CATCH1 can be manually selected with two four bit rotary switches on the board which are compared to the address lines A08-A15. Hence the Base Address can be selected in the range of

%E000 0000 to %E000 FF00.

The module transfers data in the D32 mode. It responds to 'Quad Byte Transfers' which require the signals DS0*, DS1*, LWORD* and A01 to be low.

The commands controlling the CATCH1-module are encoded in the address lines A02-A07. Therefore an offset has to be added to the module's base address for each command. Most commands also require the transfer of data. There are 13 commands defined for the CATCH1 which perform the following actions:

%00	READ_ID	reads a unique board identifier
%08	READ_FIFO	reads front-end data from the FIFO buffer
%10	READ_STATUS	reads the status of the board's components
%18	READ_FPGA	reads scalers of the Test Pulse Controller
%40	WRITE_SERIAL	writes front-end setup data to the serial transmitter
%48	WRITE_FPGA	writes test pattern or control data to the Test Pulse Controller
%50	WRITE_TRIG	writes USER1-4 to the USER Command Coding Unit
%80	PROG_FIFO	sets the programmable flags of the FIFO buffer
%88	PROG_FPGA	sets the pins for programming the FPGA
%90	BISTEN	enables the **B**uilt **I**n **S**elf **T**est of the serial data receiver
%98	REFRAME	toggles the REFRAMING option of the serial data receiver on/off
%C0	RESET_FIFO	resets the FIFO buffer (ADDRESS ONLY cycle)
%C8	RESET_FPGA	Soft reset of the Test Pulse Controller (ADDRESS ONLY cycle)

In the following these commands will be described in more detail. An overview of all commands, their offset addresses and the data transferred can be found in Table 5 at the end of this chapter. Chapter 4 contains a brief description of all the components on the CATCH1.

The board identifier is of the form "CA1000XX" (hexadecimal) where the XX stands for the serial number of the CATCH1. The READ_ID command has an offset address of %00.

The offset address of the READ_STATUS command is %10. It reads nine bit (VME data lines D15 to D07) of status information. The different bits have the following meaning:

bit D15:	INIT pin is 1 if the Test Pulse Controller is programmed, 0 when an error occurred during configuration
bit D14:	DONE pin is 1 if the Test Pulse Controller is programmed
bit D13:	EF pin is 0 if the FIFO is empty
bit D12:	PAE pin is 0 if the FIFO is almost empty (programmable)
bit D11:	PAF pin is 0 if the FIFO is almost full (programmable)
bit D10:	FF pin is 0 if the FIFO is full
bit D09:	RVS pin is 1 if transmission errors occurred (HOTLink) this information is also included in the data stream, here it is read out asynchronously
bit D08:	RDY pin is 0 if the HOTLink-self-test has finished
bit D07:	is 1 if the serial transmitter (setup data) is busy

The FIFO buffer can be cleared with the command RESET_FIFO. It has the offset address %C0 and causes an address only cycle which requires no data to be transferred. The same applies to the command RESET_FPGA with the offset address %C8. This command clears all registers of the Test Pulse Controller and puts it in the initial state after configuration.

2.2 HOTLink serial receiver

The HOTLink is a high-speed serial link. On the CATCH1 it runs with a 25 MHz (exchangeable) clock, which is the reference for a phase locked loop (PLL) that generates the high-speed transmission clock, allowing the transmission taking place at 250 MHz.

Blocks of the incoming 10 bit-coded data stream are decoded into words of one byte. These can be either data characters or special characters. For instance the special character K28.5 ('comma') is a separator for the data words. If the receiver decodes a K28.5 its internal free-running bit counter is synchronously reset on the correct byte boundaries.

All data and command words are written into the FIFO with one exception. If consecutive K28.5 characters are received, only the first one is written into the FIFO. For violations (transmission errors) and special commands the HOTLink has two bits that are combined by or into the ninth FIFO bit to identify corrupted data. The command-feature of the HOTLink is not used on the front-end board. To avoid erroneous framing on wrong byte boundaries the REFRAME command (offset address %98) can be used. Certain data sequences can be taken for a special K28.5 character and could cause corrupted data. When reframing is switched off the HOTLink receiver will not try to reframe the incoming serial data whenever it decodes such a special character.

Another command is BISTEN with the offset address %90. With this command the receiver enters a built-in self test state. For a successful self test of the transmitter-receiver system it is necessary to start the test function on the front-end side too. For example this could be done with the User command (USER4).

2.3 FIFO Buffer

The FIFO buffer is nine bit wide and can hold up to 8 KByte of data. It has four flags which indicate that it is full, almost full, almost empty or empty. By default the almost full flag is asserted when the FIFO has space for up to seven more words. On the other hand the almost empty flag is asserted when it contains seven or less words. The user can alter the value at which a flag should be asserted with the PROG_FIFO command. It has the offset address %80 and transfers 32 bit of data into the FIFO's programmable flag registers. These 32 bit are split into 4 bytes:

bits D31 to D24:	empty offset LSB register (default value: %07h)
bits D23 to D16:	empty offset MSB register (default value: %00h)
bits D15 to D08:	full offset LSB register (default value: %07h)
bits D07 to D00:	full offset MSB register (default value: %00h)

All eight bit are written into the least significant bit registers (LSB) whereas only the five least significant bits are written into the most significant bit registers. Therefore the user can program the almost empty and the almost full flags in the range from 0 to 7905.

Front-end data can be read from the FIFO using the READ_FIFO command with the offset address %08. Each time three words of eight bit are read in the big-endian format. The first word read from the FIFO is transferred in bits 23 to 16, the second in bits 15 to 8 and the third in bits 7 to

0. For each word two more bits are read out: the first is the 'ninth FIFO-bit' which indicates possible transmission errors of the incoming serial data stream. The second one is the empty flag of the FIFO buffer. In case of transmission errors the error bit is one and when the FIFO buffer is empty the bit containing the empty flag is zero. Reading data from an empty FIFO buffer results in getting always the same datum, i.e. the last datum that was written into the buffer. The bit assignment for the READ_FIFO command can be found in Table 2.

Table 2: Bit assignment for the front-end data read from the FIFO buffer

31	30 29 28	27	26 25 24	23 .. 16	15 .. 8	7 .. 0
0	Flags	0	Error Bits	Data1 (8)	Data2 (8)	Data3 (8)
	↑ ↑ ↑		↑ ↑ ↑	bits belong to data word 1 bits belong to data word 2 bits belong to data word 3		

2.4 Serial Transmitter (10 Mbit/s)

The serial interface for the front-end setup data can be selected with the WRITE_SERIAL command with the offset address %80. The interface works unidirectional and provides no error handling. Each transmission consists of two start bits, 24 data bits and two stop bits. Both the start and the stop bits are a sequence of one-zero. After data transmission the interface stays at zero.

This serial transmitter runs with a frequency of 10 MHz which is derived from the 40 MHz clock. The serial data is transmitted to the front-end system synchronously with the 40 MHz clock. This clock can be used to decode the serial data on the front-end side and to synchronize the front-end equipment in the experiment.

2.5 Test Pulse Controller

The Test Pulse Controller is implemented in a programmable logic device (FPGA) which has to be configured after each power up. This configuration process can be either done from a PC via a connector on the front panel of the CATCH1 or via the VMEbus.

The programming via the VMEbus is done using the PROG_FPGA command with the offset address %88. Depending on the datum written to this

address, three programming pins of the FPGA can be enabled, disabled, set or reset. These are the 'PROG pin' (initializes the FPGA), the 'DIN pin' (here the serial configuration data is applied) and the 'CCLK pin' (is a clock for the data). Two more pins ('INIT' and 'DONE') can be read out with the READ_STATUS command. They indicate whether the configuration was finished successfully. An example routine, written in C-language, that configures the Test Pulse Controller via VME is given in Appendix B. The different words transferred with PROG_FPGA and their results are described below:

Data lines D08 to D00	
%01	enable programming via VME
%02	disable programming via VME (default)
%04	set PROG pin to 0
%08	set PROG pin to 1
%10	set DIN pin to 1
%20	set DIN pin to 0
%40	set CCLK pin to 1
%80	set CCLK pin to 0

Once the controller is configured, 16 bit of data can be either written to or read from the Test Pulse Controller. Two commands, WRITE_FPGA with the offset address %48 and READ_FPGA with the offset address %18 can be used for this data and command transfer via the VME data lines D31 to D16.

The Test Pulse Controller knows five different commands which are encoded in the three least significant data lines D18 to D16 and its default mode is to wait for a command. A list of all possible data transfers is given in Table 3. The first command is SETUP_TRIGGER. Here the user can write all the information to select the delay and the source of the trigger signals: disable USER Command Coding Unit (see Chapter2.6) prevents USER commands from being sent to the front-end board. Automatic trigger generation causes the USER Command Coding Unit to send a trigger signal to the front-end board for each Test Pattern Strobe that was received either via VME or from the front panel. The latency is programmable: $t = t_0 + (n \cdot 25)$ ns . The offset t_0 is between 975 ns and 1000 ns. The multiplier n can be selected in the range from 0 to 27. As will be explained in Chapter 2.6 the USER commands arrive at the USER Command Coding Unit asynchronously. Hence there is an uncertainty of 25 ns (one 40 MHz clock cycle) for the timing of the USER commands. Five LEDs on the front panel indicate whether internal (red) or external (green) USER commands and Test Pattern Strobe will be used.

With the PREPARE_SCALERS command the user can do two things: he can prepare a scaler for readout and he can select which scaler(s) he wants to clear. Scalers can be prepared for readout and cleared at the same time but only after this command the scaler can be read out with READ_SCALER. Because these commands belong together they are treated as being one command in Table 3.

Table 3: List of the commands for the Test Pulse Controller

MODE	**BITS**	**VALUE**	**COMMAND**
1 : WRITE	D18-D16	%00	**SETUP_TRIGGER**
	D21	%01	disable USER Command Coding Unit
	D26-D22	%00-%1A	delay between pattern and trigger in multiples of 25 ns
		%1F	disable automatic trigger
	D31-D27	1	enable external USER signals,
		0	disable external USER signals: Test Pattern Strobe (D31) User (D30), Clear (D29), Reset (D28) and Trigger input (D27)
2a: WRITE	D18-D16	%01	**PREPARE_SCALERS**
	D26-D24	%00-%03	for Trigger, Reset, Clear, User
		%04	for Test Pattern Strobe
	D31-D27	%01-%1F	clear scalers for pattern strobe (D31), User (D30), Clear (D29), Reset (D28) and Trigger (D27)
2b: READ	D31-D16		**READ_SCALER**
3 : WRITE	D18-D16	%02	**SET_DAC**
	D31-D19		12 bit DAC setup value
4a: WRITE	D18-D16	%03	**WRITE_PATTERN**
4b:	D31-D16		16 bit test pattern 01-16
4c:			16 bit test pattern 17-32
4d:			16 bit test pattern 33-48
4e:			16 bit test pattern 49-64
5 : WRITE	D18-D16	%04	**PATTERN_STROBE**

The third command is SET_DAC which programs the digital to analog converter on the 'Pulser Box'. After the command WRITE_PATTERN the FPGA waits for another four data transfers which contain the 64 bit test

pattern for the front-end. When all 64 bit are received they can be sent to the 'Pulser Box' with the completing command PATTERN_STROBE. This command, executed without new test pattern information, will transfer the most recent test pattern data to the 'Pulser Box'.

The seven segment display on the front panel of the CATCH1 shows the actual mode of the Test Pulse Controller and possible errors. Once an error occurred its value stays on the display until reset or another error occurs. A list of all values is given in Table 4.

Table 4: The 7-Segment display shows the status of the Test Pulse Controller

Display	STATUS
-	configuration error / not configured
.	controller waits for more data or READ_SCALER command
0	no error
1	WRITE command expected
2	READ_SCALER expected
3	WRITE_PATTERN data word 1, 2 or 3 expected
4	WRITE_PATTERN data word 4 expected
5	unknown command

2.6 User Command Coding Unit

The USER Command Coding Unit uses the 40 MHz clock to encode four different USER commands (Trigger, Reset, Clear and User) in terms of clock cycles before they are transmitted to the front-end board via the S-UTP cable. The shortest signal is the Trigger with a length of one clock cycle (25 ns) while the Reset, the Clear and the User commands have lengths of 50 ns, 75 ns and 100 ns respectively.

During normal operation the Test Pulse Controller sends an automatically generated trigger signal to the USER Command Coding Unit for each test pattern that was written to the 'Pulser Box'. Delay between Test Pattern Strobe and the trigger signal can be adjusted in multiples of 25 ns plus an offset of 975 ns in the Test Pulse Controller.

In case the automatic trigger generation is disabled, the trigger signals can either come from the VMEbus or from the ECL input connector on the front panel. These are the only possibilities to feed the other three USER commands and the pattern strobe into the CATCH1. The five ECL channels

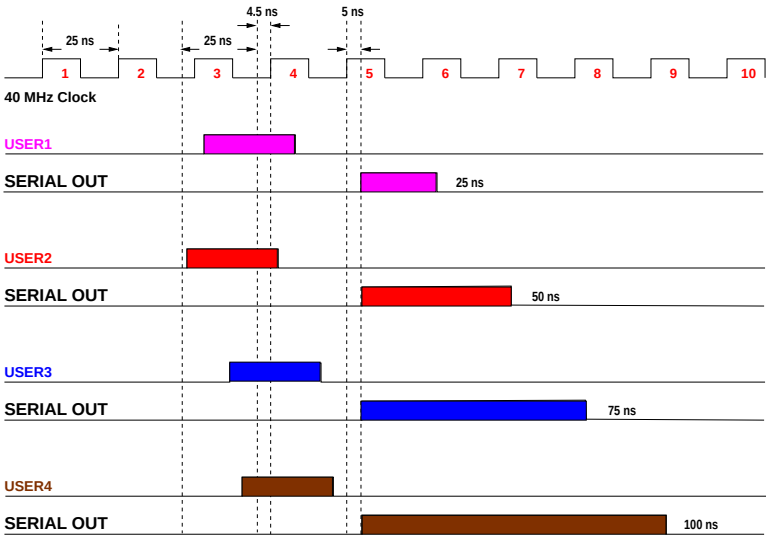


Figure 3: USER command timings

on the connector can be individually enabled with the SETUP_TRIGGER command (see Table 3). The applied signals must be at least 30 ns long. After one signal has been encoded the next one can be handled one clock cycle later. Maximum rates for the USER commands can be found in Chapter 5.2.

The timing of the outgoing USER commands with respect to the 40 MHz clock can be taken from Figure 3. Because the incoming signals (from VME, automatically generated Triggers or ECL signals) arrive asynchronously at the USER Command Coding Unit, the encoded USER commands have an uncertainty of one clock cycle. In Figure 3 incoming signals that are present in the 25 ns interval ending 4.5 ns before the rising edge of clock cycle 4 produce encoded signals with the same timing. The 4.5 ns are the chip's internal register setup time. For incoming signals that are present within this time before a rising edge of the clock it is not predictable if the encoding will start with the next rising edge or only one clock cycle later.

The USER command transfer via VME is done with the WRITE_TRIG command that has the offset address %50. Here the signals are sent via the VME data lines D03 to D00 which correspond to the User, the Clear, the Reset and the Trigger signals respectively. Whenever two USER commands are applied at the same time, the shorter output signal has priority. For instance only the trigger signal will be encoded if both the trigger and the reset signal are applied to the coding unit. However this should not be done via the VME command WRITE_TRIG. If necessary the trigger coding unit can be disabled from the Test Pulse Controller (see Table 3).

Table 5: Offset addresses of the CATCH1 commands

COMMAND	OFFSET	DATA WORD	
READ_ID	%00	D31-D00	CATCH1 identifier
READ_FIFO	%08	D31-D00	three data words
READ_STATUS	%10	D15-D07	status bits
READ_FPGA	%18	D31-D16	scaler values
WRITE_SERIAL	%40	D23-D00	setup data to front-end
WRITE_FPGA	%48	D31-D16	pattern, pulse height
WRITE_TRIG	%50	D03=1	User command
		D02=1	Clear command
		D01=1	Reset command
		D00=1	Trigger command
PROG_FIFO	%80	D31-D16	set FIFO flags
PROG_FPGA	%88	D07-D00:	data transferred:
		%01	enable programming pins
		%02	disable programming pins
		%04	set PROG pin to 0
		%08	set PROG pin to 1
		%10	set DIN pin to 1
		%20	set DIN pin to 0
		%40	set CCLK pin to 1
		%80	set CCLK pin to 0
REFRAME	%98	D07-D00	
		%01	enable reframing
		%02	disable reframing
BISTEN	%90	D07-D00	
		%01	start built in self test
		%02	stop built in self test
RESET_FIFO	%C0	-	reset FIFO buffer
RESET_FPGA	%C8	-	reset Test Pulse Controller

3 Board Layout

3.1 Front Panel Information

Figure 4 shows a sketch of the CATCH1 front panel and a table describing the function of its LEDs, connectors and buttons.

3.2 Front Panel Pin Assignment

There are four connectors on the CATCH1 front panel. The uppermost is a RJ-45 for the communication with the front-end board via a S-UTP cable (CAT 5+). While the high-speed data transfer via HOTLink (PECL), the USER commands (LVDS) and the 40 MHz clock are differential signals, the serial setup data is generated as a single ended signal (TTL). However because an opto-coupler will be used to receive this signal on the front-end board, the cable carries both lines to drive the LED of the opto-coupler.

Table 6: RJ-45 Connector

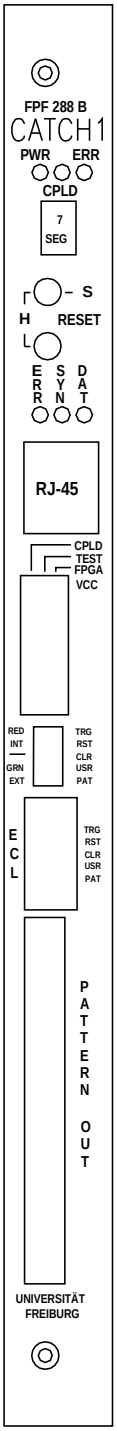
PIN #	SIGNAL NAME
1	40 MHz Clock (-)
2	40 MHz Clock (+)
3	Serial Setup Data (-)
4	USER 1-4 (+)
5	USER 1-4 (-)
6	Serial Setup Data (+)
7	HOTLink Data (-)
8	HOTLink Data (+)

Table 7: Programming Connectors

PIN #	'CPLD'	'TEST'	'PROG'
9	V_{CC}	RT	V_{CC}
8	GND	RD	GND
7	n.c.	TRIGG	n.c.
6	TCK	n.c.	CCLK
5	n.c.	TDI	DONE
4	TDO	TCK	DIN
3	TDI	TMS	/PROG
2	n.c.	CLKI	/INIT
1	TMS	CLKO	/SOFTRES

Table 8: ECL Connector

Signal Name	PIN #		Signal Name
USER 1 (+)	10 •	• 09	USER 1 (-)
USER 2 (+)	08 •	• 07	USER 2 (-)
USER 3 (+)	06 •	• 05	USER 3 (-)
USER 4 (+)	04 •	• 03	USER 4 (-)
Pattern Strobe (+)	02 •	• 01	Pattern Strobe (-)



BUTTON	DESCRIPTION
RESET	S clears all registers of FPGA (soft reset) and both S and H put the CATCH1 power-up state. (hard reset)

7 SEGMENT	DESCRIPTION
	status of the Test Pulse Controller. For more details see Table 4 in Chapter 2.5

LED	DESCRIPTION
PWR (green)	power
CPLD (green)	VME interface is busy
ERR (red)	FPGA error during configuration
ERR (red)	corrupted data from front-end
SYNC (green)	synchronization characters received
DATA (yellow)	serial data arrives from the front-end
TRG (red/green)	Trigger
RST (red/green)	Reset
CLR (red/green)	Clear
USR (red/green)	User
PAT (red/green)	Pattern Strobe
	red: internal - green: external ECL

CONNECTOR	DESCRIPTION
RJ-45	carries serial HOTLink data, USER commands, the 40 MHz clock (all differential) and serial front-end setup data. Setup data and USER commands are transmitted synchronously to the 40 MHz clock.
CPLD	coded connector to program the CPLD
TEST	coded connector to debug the FPGA
FPGA	coded connector to program the FPGA
ECL	ECL inputs for the trigger signals (+ -)
PATTERN OUT (Robinson Nugent)	TTL test pattern transfer with 64 pins test pattern (including 12 DAC value), 1 pin DAC control, 1 pin trigger and 10 pins GND.

Figure 4: The Front Panel of the CATCH1

Table 9: Robinson Nugent Connector

SIGNAL NAME	PIN NUMBER		SIGNAL NAME
GND	B40 •	• A40	GND
GND	B39 •	• A39	GND
DACSTR(+)	B38 •	• A38	DOUTSTR(+)
DACSTR(-)	B37 •	• A37	DOUTSTR(-)
GND	B36 •	• A36	GND
GND	B35 •	• A35	GND
not used	B34 •	• A34	GND
not used	B33 •	• A33	GND
DOUT63	B32 •	• A32	DOUT62
DOUT61	B31 •	• A31	DOUT60
DOUT59	B30 •	• A30	DOUT58
DOUT57	B29 •	• A29	DOUT56
DOUT55	B28 •	• A28	DOUT54
DOUT53	B27 •	• A27	DOUT52
DOUT51	B26 •	• A26	DOUT50
DOUT49	B25 •	• A25	DOUT48
DOUT47	B24 •	• A24	DOUT46
DOUT45	B23 •	• A23	DOUT44
DOUT43	B22 •	• A22	DOUT42
DOUT41	B21 •	• A21	DOUT40
DOUT39	B20 •	• A20	DOUT38
DOUT37	B19 •	• A19	DOUT36
DOUT35	B18 •	• A18	DOUT34
DOUT33	B17 •	• A17	DOUT32
DOUT31	B16 •	• A16	DOUT30
DOUT29	B15 •	• A15	DOUT28
DOUT27	B14 •	• A14	DOUT26
DOUT25	B13 •	• A13	DOUT24
DOUT23	B12 •	• A12	DOUT22
DOUT21	B11 •	• A11	DOUT20
DOUT19	B10 •	• A10	DOUT18
DOUT17	B09 •	• A09	DOUT16
DOUT15	B08 •	• A08	DOUT14
DOUT13	B07 •	• A07	DOUT12
DOUT11	B06 •	• A06	DOUT10
DOUT09	B05 •	• A05	DOUT08
DOUT07	B04 •	• A04	DOUT06
DOUT05	B03 •	• A03	DOUT04
DOUT03	B02 •	• A02	DOUT02
DOUT01	B01 •	• A01	DOUT00

The differential signals should be decoupled by capacitors on the front-end board. Table 6 shows the pin assignment for the RJ-45 connector.

Three connectors are foreseen for programming and debugging of the FPGA and the CPLD devices. With the 'CPLD' JTAG connector the VME interface can be reprogrammed. 'PROG' and 'TEST' can be used to program and debug the FPGA. Table 7 shows the pin assignments.

The external signals USER1 to USER4 and Test Pattern Strobe can be applied to the ECL connector. The pinning is listed in Table 8

The test pattern output is an 80 pin Robinson Nugent connector. This connector is used to transfer the TTL test pattern to the 'Pulser Box'. The 64 bit test pattern are transferred via lines D00 to D63.

Optionally lines D00-D11 can carry the 12 bit needed to set the DAC. This information is loaded into the DAC with the differential signal DACSTR. DOUTSTR is the second differential signal enabling the analog output of the 'Pulser Box'. Table 9 shows the pin allocation for the signals.

3.3 Rear Panel Pin Assignments

Tables 10 and 11 show the pin assignment for the VME signals used on the CATCH1 board.

3.4 Jumper Settings

Five different clock speeds can be selected for the programmable logic devices on the CATCH1 board: 8 and 17 MHz are derived from an exchangeable 33 MHz clock and 25 MHz are derived from an exchangeable 50 MHz clock. These clocks can be individually selected for both chips by shorting the appropriate jumpers JP4 (FPGA) and JP5 (CPLD). For the FPGA there is also a 80 MHz clock foreseen which can only be selected by the logic implementation in this device and is unused in the current design. Figure 5 shows the location of the jumpers for clock selection on the board. The default setting are 33 MHz for the CPLD as well as for the FPGA.

Besides the internal clocks on the CATCH1 an external one can be selected for the FPGA. This external clock is enabled with Jumper JP3 and can be taken from a PC. Selecting external clocks should only be done for debugging the FPGA. During normal operation the two INT pins are shorted with a jumper by default (see Figure 5).

The remaining two jumpers also serve debugging purposes. If JP1 is shorted the user can send trigger signals via the 'TEST' connector to the FPGA. A shorted jumper JP2 allows soft reset signals to be sent from a PC

Table 10: VME J1/P1 Connector

PIN #	ROW A	ROW B	ROW C
1	D00	n.c.	D08
2	D01	n.c.	D09
3	D02	n.c.	D10
4	D03	n.c.	D11
5	D04	n.c.	D12
6	D05	n.c.	D13
7	D06	n.c.	D14
8	D07	n.c.	D15
9	GND	n.c.	GND
10	n.c.	BG3IN*	n.c.
11	GND	BG3OUT*	BERR*
12	DS1*	n.c.	SYSRESET*
13	DS0*	n.c.	LWORD
14	WRITE*	n.c.	AM5
15	GND	n.c.	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	n.c.	A17
22	IACKOUT*	n.c.	A16
23	AM4	GND	A15
24	A07	n.c.	A14
25	A06	n.c.	A13
26	A05	n.c.	A12
27	A04	n.c.	A11
28	A03	n.c.	A10
29	A02	n.c.	A09
30	A01	n.c.	A08
31	-12 V	n.c.	n.c.
32	+5 V	+5 V	+5

Table 11: VME J2/P2 Connector

PIN #	ROW A	ROW B	ROW C
1	n.c.	+5 V	n.c.
2	n.c.	GND	n.c.
3	n.c.	n.c.	n.c.
4	n.c.	A24	n.c.
5	n.c.	A25	n.c.
6	n.c.	A26	n.c.
7	n.c.	A27	n.c.
8	n.c.	A28	n.c.
9	n.c.	A29	n.c.
10	n.c.	A30	n.c.
11	n.c.	A31	n.c.
12	n.c.	n.c.	n.c.
13	n.c.	+5 V	n.c.
14	n.c.	D16	n.c.
15	n.c.	D17	n.c.
16	n.c.	D18	n.c.
17	n.c.	D19	n.c.
18	n.c.	D20	n.c.
19	n.c.	D21	n.c.
20	n.c.	D22	n.c.
21	n.c.	D23	n.c.
22	n.c.	n.c.	n.c.
23	n.c.	D24	n.c.
24	n.c.	D25	n.c.
25	n.c.	D26	n.c.
26	n.c.	D27	n.c.
27	n.c.	D28	n.c.
28	n.c.	D29	n.c.
29	n.c.	D30	n.c.
30	n.c.	D31	n.c.
31	n.c.	n.c.	n.c.
32	n.c.	+5 V	n.c.

to the FPGA via the 'PROG' connector on the front panel. There is no need to change these settings during normal operation.

3.5 Rotary Switches

Two rotary switches adjust the base address of the CATCH1 board in the range of %E0000000 to %E000FF00. Switch SW2 selects the lower four bit and switch SW1 the upper four bit of the base address. For the location on the board see Figure 5.

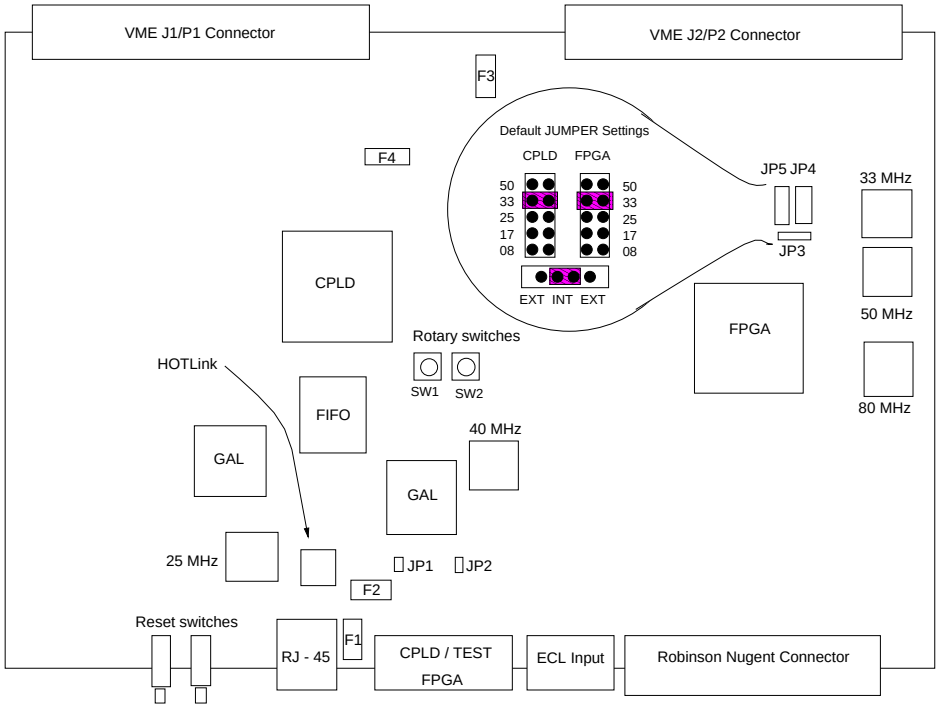


Figure 5: Components on the CATCH1

4 Components on the Board

In this section a brief overview will be given on the important ICs on the CATCH1 board. Their locations can be taken from Figure 5 and detailed descriptions can be found in the corresponding data sheets or specifications.

4.1 Xilinx XC95288-15 CPLD

This 'Complex Programmable Logic Device' (CPLD) houses the VME interface logic and the serial transmitter for the setup data to the front-end. It retains its information when power is switched off but can be reprogrammed via the 'CPLD' JTAG connector on the front panel. For detailed information see [1].

4.2 Xilinx XC4020E-2 FPGA

This IC is a 'Field Programmable Gate Array' in which the Test Pulse Controller is implemented. It has to be configured on each power up. As already mentioned this can be done either via the VMEbus or via the FPGA connector on the front panel of the CATCH1. The configuration via VME requires

configuration data in form of a file that contains either binary or hexadecimal numbers. This data will be transferred to the chip bit by bit. An example routine, written in C-language, can be found in Appendix B.

Since the FPGA can be easily and quickly reconfigured it can perform quite different tasks. For instance it could be used to read data into the CATCH1 from the Robinson Nugent connector and thus use the FPGA for digitization. For a detailed description of the XC4020E-2 see [1].

4.3 Cypress CY7B933 HOTLink Receiver

The HOTLink receives data from the transmitter on the front-end side. Both the transmitter and the receiver side must have a reference clock which must be within $\pm$ 0.1% to each other. These clocks are exchangeable and can operate at different frequencies up to 33 MHz. A detailed description of the HOTLink transmitter and receiver devices can be found in [2].

4.4 CY7C4251-15 JC FIFO

This FIFO buffer is nine bit wide and 8 KByte deep. It features empty, almost empty, almost full and full flags of which the almost empty/full flags can be programmed. The FIFO belongs to a family of pin-compatible buffers and can be easily replaced in case a bigger memory is required. A detailed description can be found in [3].

4.5 Lattice GAL22V10C GAL

The USER Command Coding Unit and part of the interface logic between HOTLink and FIFO buffer are each implemented in a GAL ('Generic Array Logic'). They are exchangeable and can be reprogrammed. However, these devices are not capable to hold as much logic as a CPLD. For more details see [4].

4.6 CY7C199-10vc Memory

This memory (36 KByte) can be accessed via the FPGA but is not used in the current design. For example, test pattern could be stored here before being sent to the 'Pulser Box'. A detailed description of this device is available in [5].

5 Electrical and Mechanical Specifications

5.1 Cables

The three connectors for programming and debugging the CPLD and the FPGA require special Xilinx-adapters. These have to be connected to a PC and should not be used during normal operation. The following list gives an overview of the cables for the other connectors on the front panel:

S-UTP cable - CAT5+ (shielded twisted pair)
Length : $\leq$ 20 m
Wires : 8

Twisted pair flat cable to 'Pulser Box'
Length : $\leq$ 3 m
Wires : 80

Twisted pair flat cable for ECL signals
Length : $\leq$ 3 m
Wires : 10

5.2 Signal Characteristics

Signals on the S-UTP cable

HOTLink data	:	Differential PECL level, max. frequency 330 MHz, the lines are terminated with 110 Ω to V_{CC} and with 300 Ω to GND
USER commands	:	Differential LVDS level
Clock	:	Differential LVDS level, frequency: 40 MHz
Serial setup data (-)	:	TTL level, frequency: 10 MHz
Serial setup data (+)	:	for driving the LED of an opto-coupler

Signals on the twisted pair cable (80 wires)

Data lines D00-63	:	TTL level
DACSTR	:	Differential LVDS level
DOUTSTR	:	Differential LVDS level
remaining wires	:	GND or not used

Signals on the twisted pair cable (10 wires)

All USER commands	:	ECL level, dynamically terminated with 100 Ω
Pattern Strobe	:	ECL level, dynamically terminated with 100 Ω

ECL input pulses

- width : $\geq$ 30 ns

The device driving the external ECL signals must have a 750 Ω emitter pull-down resistor and a ground connection to the CATCH1 must exist.

USER command output

- width : 25 ns to 100 ns
- USER1 (TRIGGER) frequency : $\leq$ 15 MHz (fastest)
- USER4 (User) frequency: $\leq$ 5 MHz (slowest)

Test Pulse Pattern output

- Pattern strobe width : 30 ns
- DAC data strobe width : 150 ns
- DAC data setup time : 120 ns
- DAC data hold time : 20 ns

On the CATCH1 the timing of DAC data strobe (DACSTR±) and data signals (DOUT00 - DOUT11) is adjusted to the requirements of the digital-to-analog converter used on the 'Pulser Box' (AD7845) [6]. Signal levels and timing can be taken from Figure 6 where the 12 bit DAC data (DOUT00 - DOUT11) are active high.

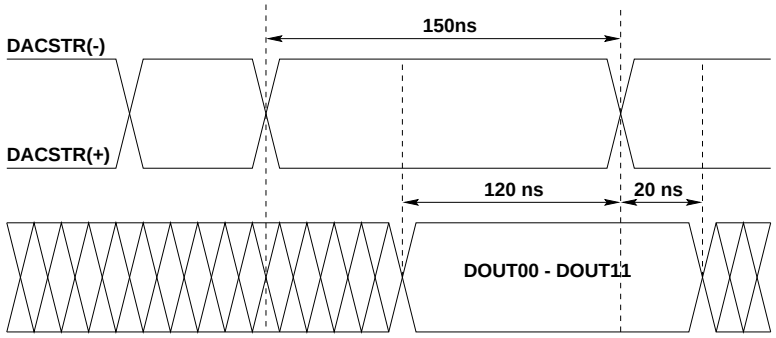


Figure 6: DAC setup timing

The 64 test pattern signals are active high and remain present at the Robinson Nugent connector until another pattern is transferred to the Test Pulse Controller. Pattern strobe is also a differential signal: DOUTSTR(+) is positive and DOUTSTR(-) negative when the test pattern is sent to the front-end board.

VME Performance (with MVME 2604)

- Writing 64 bit test pattern + pattern strobe : 0.33 MHz
- Writing only pattern strobe : 1.92 MHz
- Writing USER commands : 2.73 MHz
- Reading FIFO data : 1.11 MHz
- Reading Board Status: 1.20 MHz

5.3 Power Requirements, Mechanical Size

+5 V / 1150 mA protected with a 3 A fuse (F3)
-12 V / 25 mA protected with a 250 mA fuse (F4)

+5 V, 'CPLD' (F1) and 'PROG' (F2) connectors each protected with a 250 mA fuse.

For the location of the fuses see Figure 5.

The CATCH1 is a single-width 6U VME-module.

References

[1] Xilinx: *The Programmable Logic Databook* (1996).

[2] CYPRESS Semiconductor Corporation, *CY7B923/33 HOTLink Transmitter/Receiver* Data Sheet.

[3] CYPRESS Semiconductor Corporation, *CY7C42x1 8Kx9 Synchronous FIFO* Data sheet.

[4] Lattice Semiconductor Corporation, *Data Book* (1994)e

[5] CYPRESS Semiconductor Corporation, *CY7C199 Memory* Data sheet.

[6] Analog Devices, *AD7845 D/A Converter* Data sheet.

A Getting started with the CATCH1

The consecutive steps, necessary to run tests with the CATCH1 board are described in this Appendix:

1. switch off the power of your VME crate
2. select suitable base address(es) for your CATCH1 board(s)
3. plug in the board(s)
4. connect the 'Pulser Box(es)' and the front-end electronics
5. switch power on again
6. now change to directory where your 'CATCH1 software' is installed
7. run the configuration routine for the FPGA (file containing the design required!)
8. when the seven segment display shows a zero everything is ready; if not use the right configuration data.

B Example: FPGA Configuration

Here an example will be given to demonstrate how the logical design must be loaded on the FPGA via VME. The design must be available in form of a file containing only binary or hexadecimal numbers. Hexadecimal files generated from the Xilinx software have to be converted into binary ones. Here it is important that each two hexadecimal numbers are swapped and their bits are reversed before they are written to the VME.

```
#include<stdio.h>

void            PROG_FPGA(unsigned long VALUE);
unsigned long   READ_STATUS();
void            WAIT();

main(int argc, char *argv[])
{
  FILE *input;

  int i, a;
  char bit;
  unsigned long ENA_H, ENA_L, PROG_H, PROG_L, DIN_H, DIN_L;
  unsigned long BASE, VALUE, CCLK_H, CCLK;

  /*  some constants  */

  BASE          = 0xE0000000;
  ENA_H         = 0x00000001;     ENA_L         = 0x00000002;
  PROG_L        = 0x00000004;     PROG_H        = 0x00000008;
  DIN_H         = 0x00000010;     DIN_L         = 0x00000020;
  CCLK_H        = 0x00000040;     CCLK_L        = 0x00000080;

  PROG_FPGA(ENA_H);     /*  enable the programming pins  */

  /*  reset FPGA with the PROG pin: first low then high  */

  PROG_FPGA(PROG_L);
  PROG_FPGA(PROG_H);

  WAIT();

  /*  INIT pin is high now: FPGA is ready for programming  */
```

```
  printf("\nReading BOARD_STATUS...    ");
  VALUE = READ_STATUS(); a = (VALUE & 0x4000 ? 1 : 0);
  printf("INIT : %d\n", a);

  /*  reading the file containing the configuration data  */

  while(!feof(input)){
    fscanf(input,"%c", &bit);
    if(bit == '\r')
      break;
    else if(bit == '1'){
      PROG_FPGA(DIN_H);      /*  set the DIN pin to one  */
      PROG_FPGA(CCLK_H);     /*  assert one CCLK clock cycle  */
      PROG_FPGA(CCLK_L);
    }
    else if(bit == '0'){
      PROG_FPGA(DIN_L);      /*  set the DIN pin to one  */
      PROG_FPGA(CCLK_H);     /*  assert one CCLK clock cycle  */
      PROG_FPGA(CCLK_L);
    }
  }

  for(i = 0; i <= 15; i = i + 1){
    PROG_FPGA(DIN_H);
    PROG_FPGA(CCLK_H);    /*  16 more ones to the FPGA  */
    PROG_FPGA(CCLK_L);    /*  required to finish configuration  */
  }

  /*  check whether programming was successful (DONE pin high)  */

  printf("\nReading BOARD_STATUS...    ");
  VALUE = READ_STATUS(); a = (VALUE & 0x4000 ? 1 : 0);
  printf("DONE : %d\n", a);

  if(a == 1)
    printf("\nConfiguration cycle completed!\n\n");
  else
    printf("\nConfiguration cycle failed!\n\n");

  PROG_FPGA(ENA_L);  /*  disable programming pins  */
  exit(1);
}
```

```
/*  SUBROUTINES  */

/*  PROG_FPGA : change state of FPGA's programming pins  */
void PROG_FPGA(unsigned long VALUE)
{
  unsigned long ADDRESS;
  ADDRESS       = 0xE0000000 + 0x00000088;   /*  BASE + OFFSET  */
  * (unsigned long *) ADDRESS   = VALUE;
  return;
}

/*  READ_STATUS : check the status of the two pins INIT & DONE  */
unsigned long READ_STATUS()
{
  unsigned long ADDRESS, DATUM;
  ADDRESS       = 0xE0000000 + 0x00000010,   /*  BASE + OFFSET  */
  DATUM         = * (unsigned long *) ADDRESS;
  return DATUM;
}

/*  Wait a while  */
void WAIT()
{
  int i;
  printf("\nWAITING...");
  for(i = 0; i <= 500000; i = i + 1){}
  printf("\n");
  return;
}
```

C List of Sheets

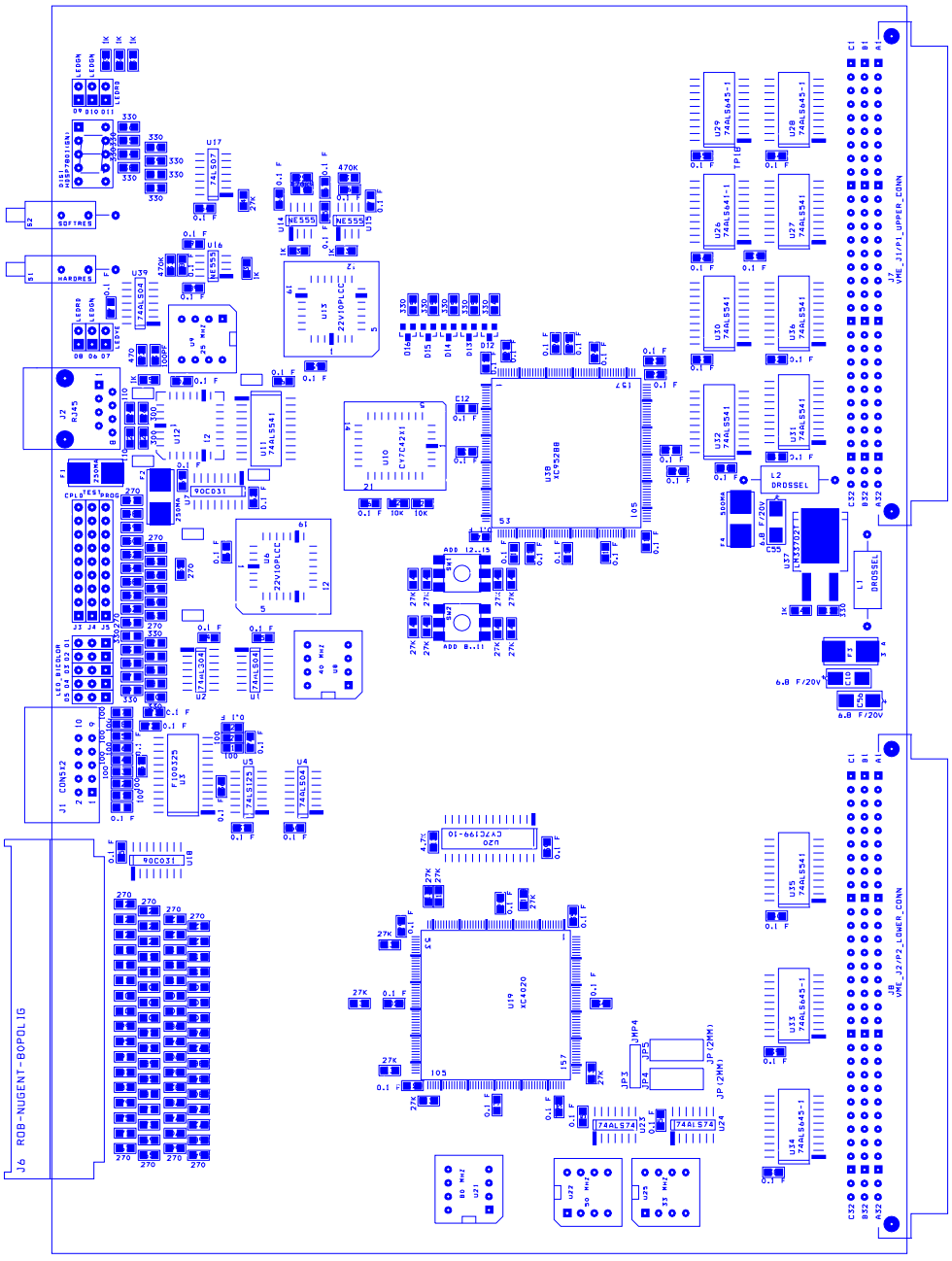


Figure 7: IC Placement

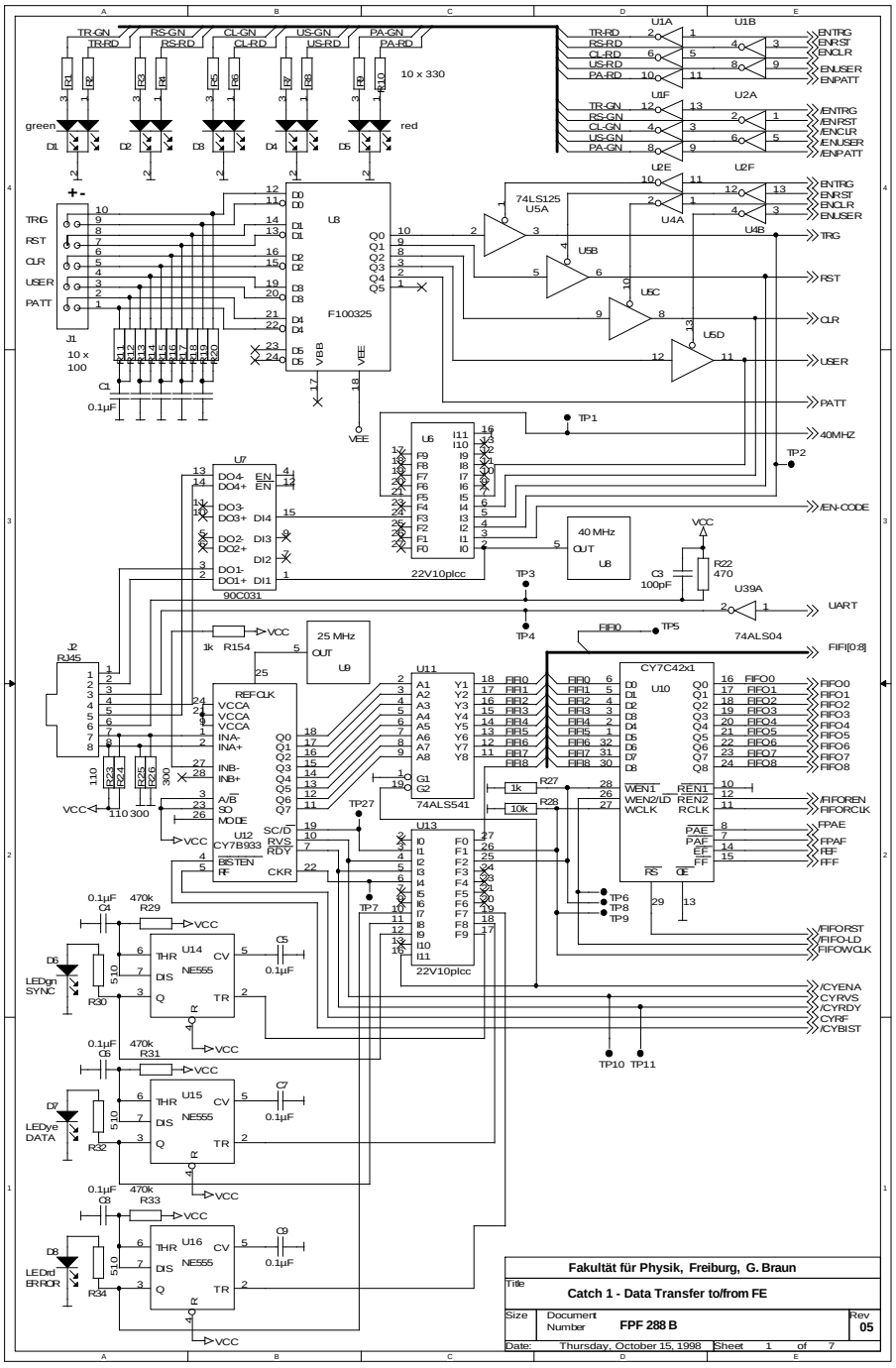


Figure 8: Data Transfer from/to FE

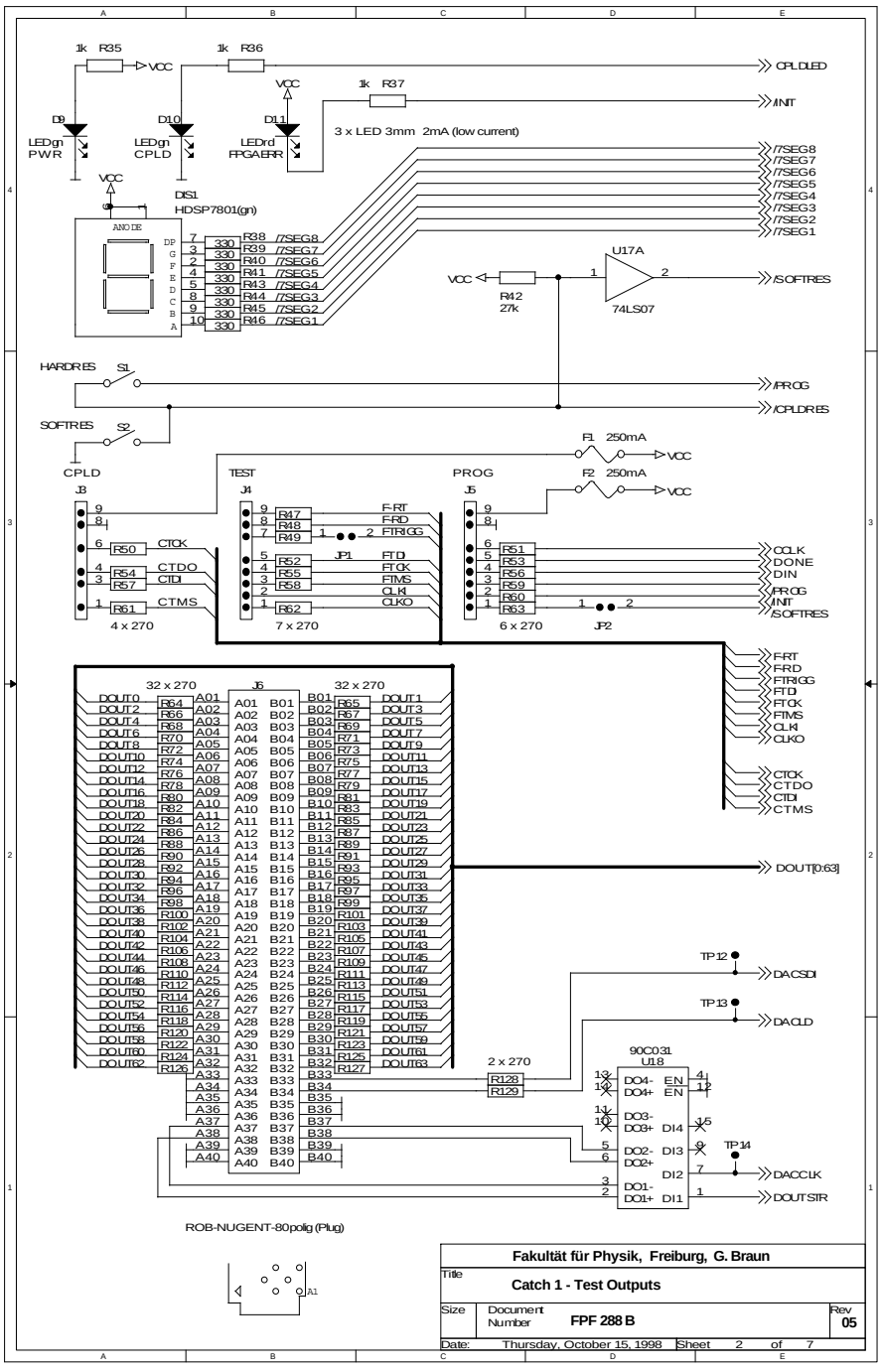


Figure 9: Test Outputs

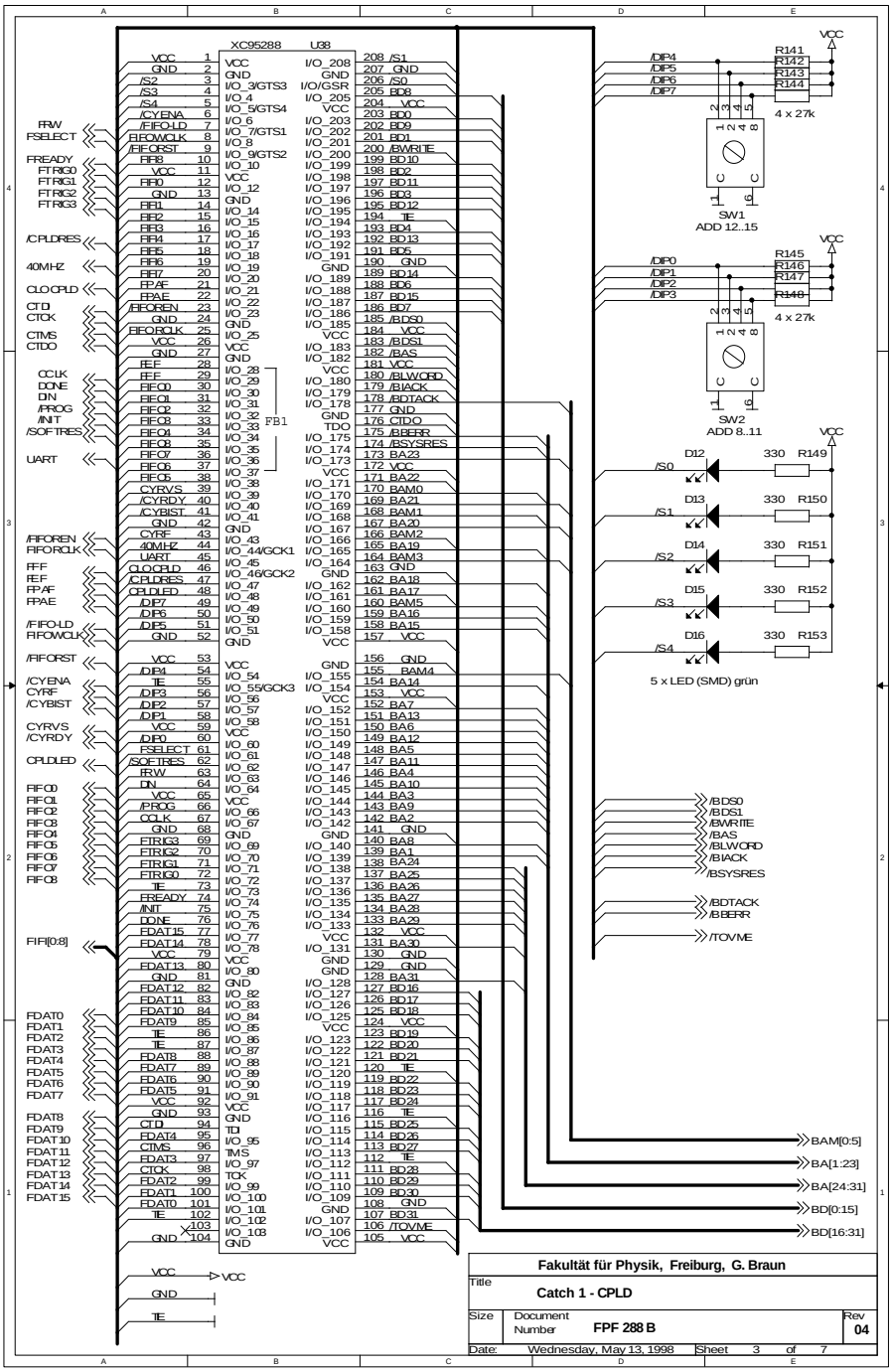


Figure 10: CPLD

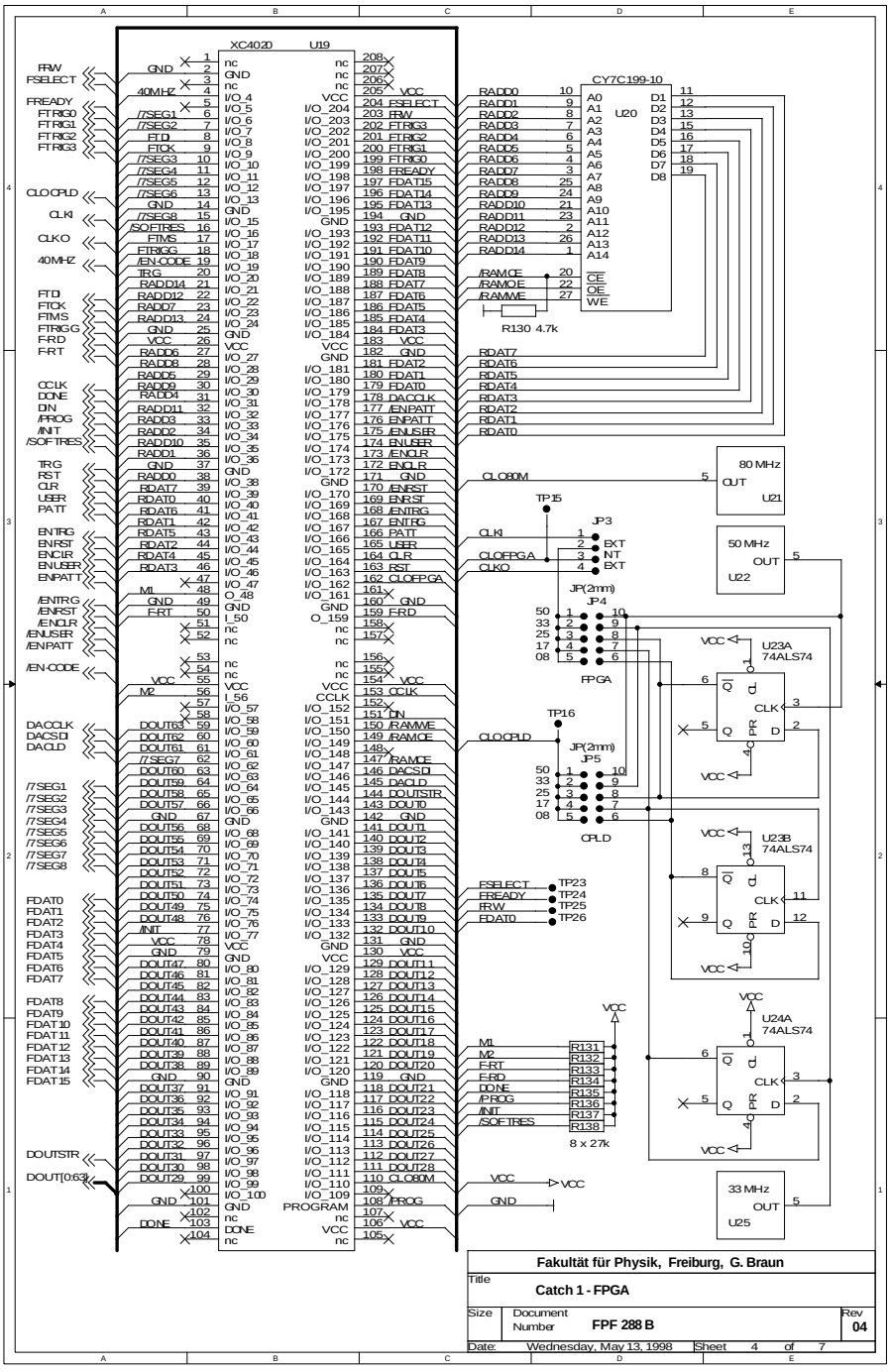


Figure 11: FPGA

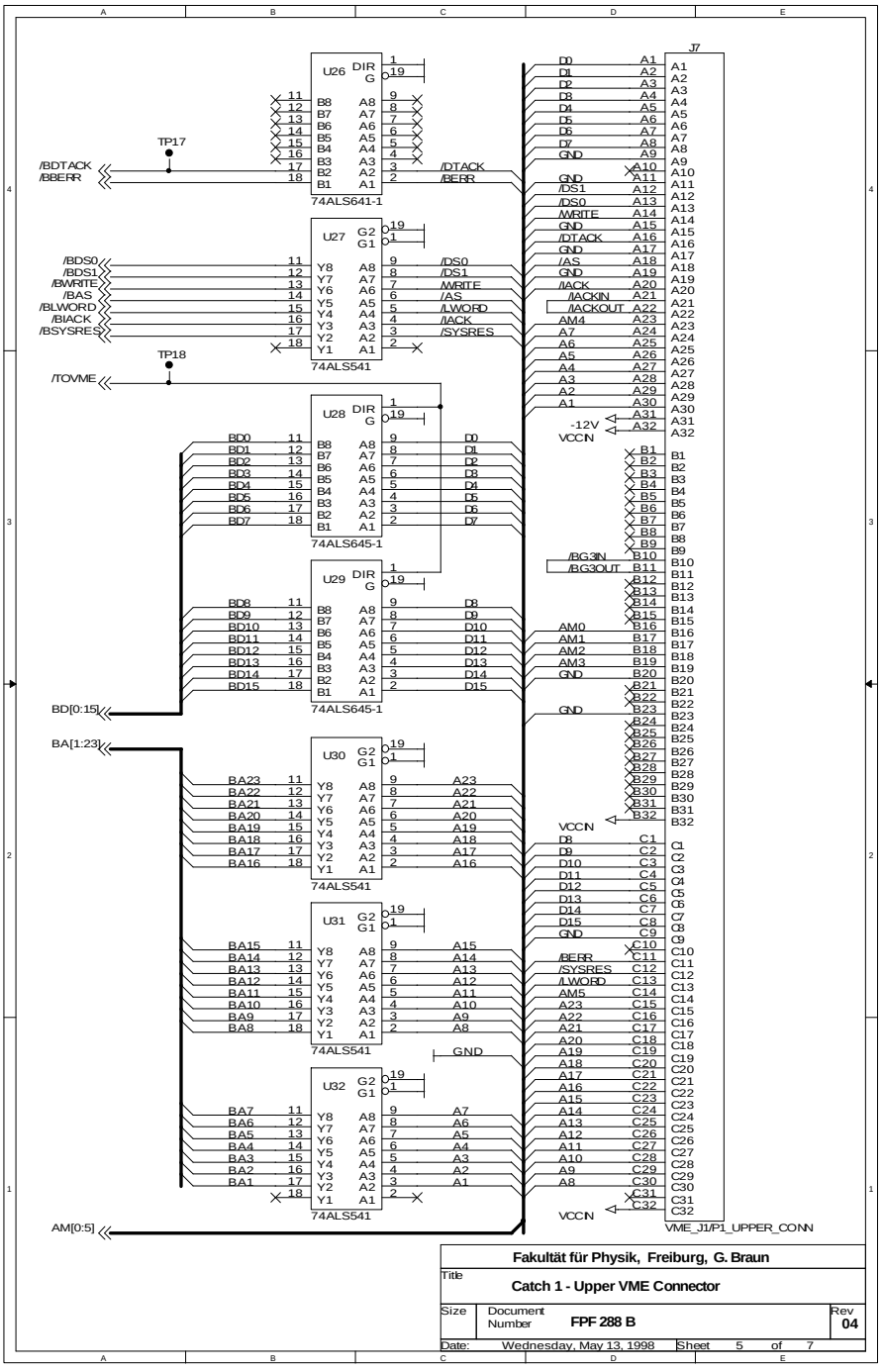


Figure 12: Upper VME connector

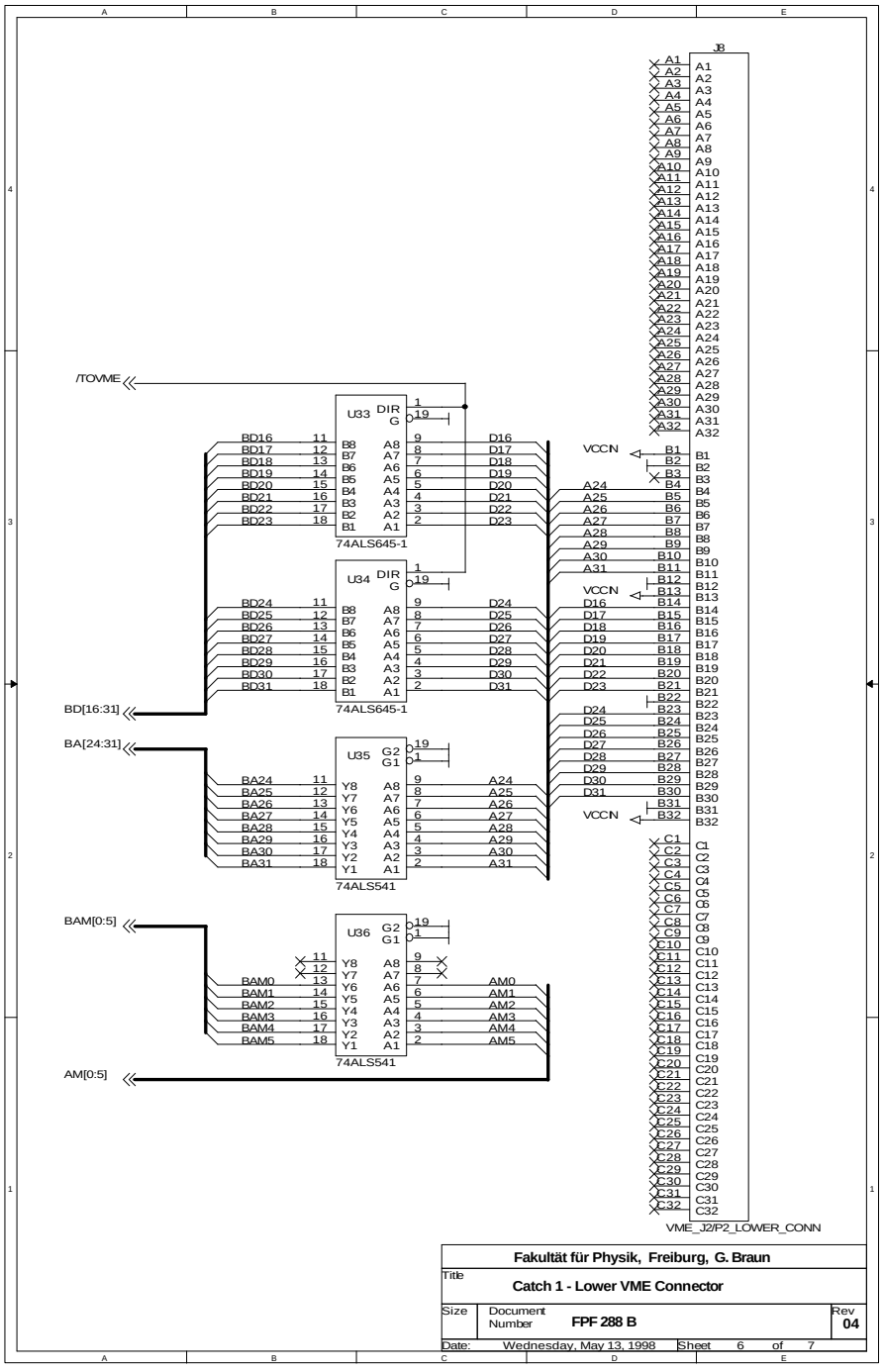


Figure 13: Lower VME connector

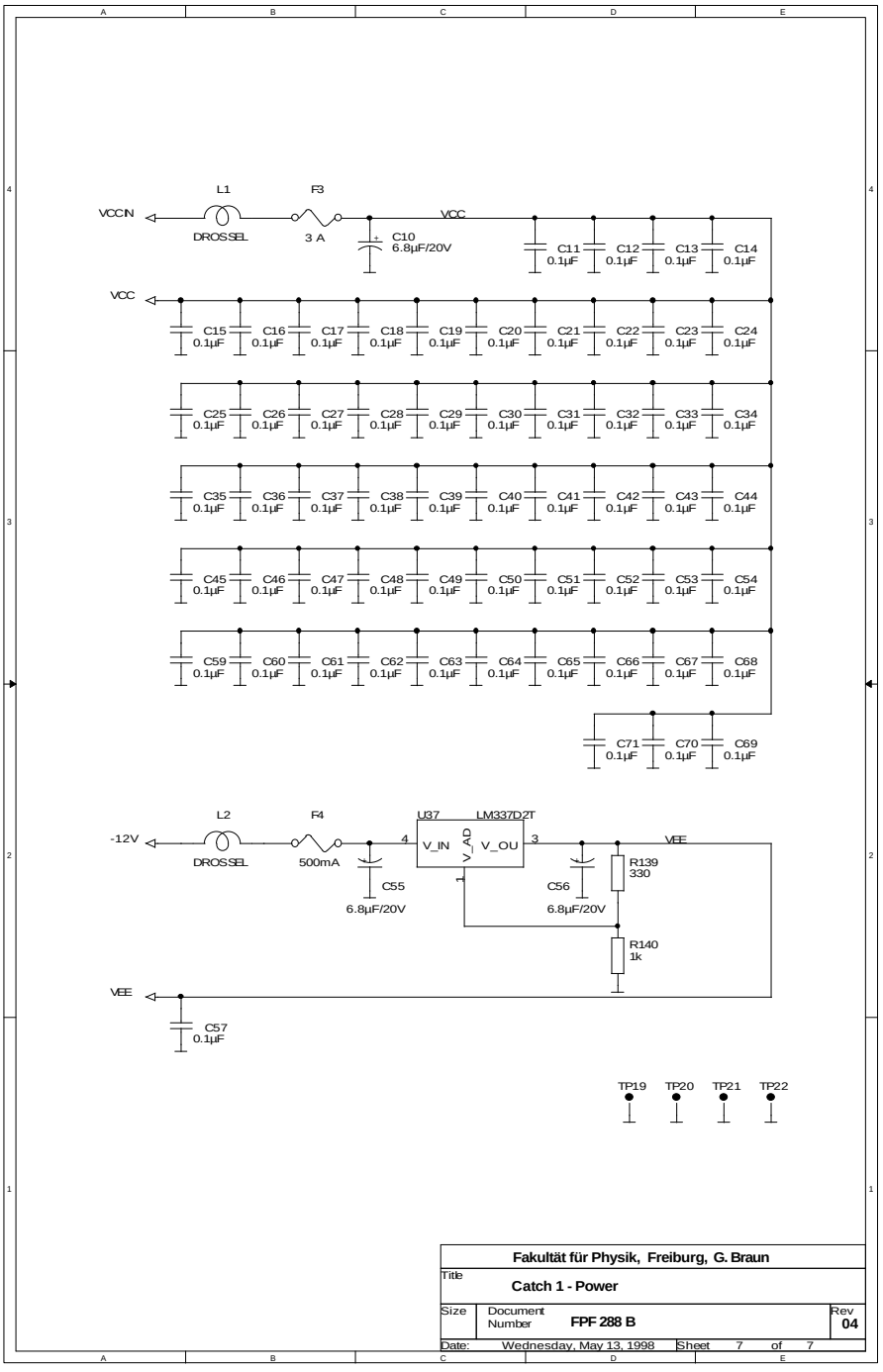


Figure 14: Power